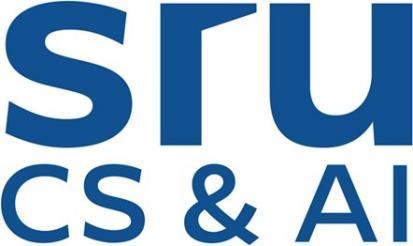
# PE1-Data Analysis Using Python



A Course Completion Report in partial fulfilment of the degree

## Bachelor of Technology

in

**ComputerScience&Artificial Intelligence**

**By**

**Roll. No :** 2203A52116 **Name**: PANCHAGIRI VIDYA SRI

**Batch No:** 31

**Submitted to**





**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE**

**SR UNIVERSITY, ANANTHASAGAR, WARANGAL**

**March, 2025.**

**Mobile Phone Usage in India (csv)-dataset-1**

**Abstract**

This study analyses mobile phone usage patterns across different age groups, regions, and user behaviours in India. The dataset includes attributes such as screen time, app preferences, purpose of use, and demographic details. Using statistical analysis and machine learning classification models, the project aims to uncover trends, predict excessive usage, and assess the impact of mobile habits on lifestyle. The insights gained can support initiatives in digital well-being, targeted awareness campaigns, and policy-making around responsible mobile usage in one of the world’s fastest-growing digital markets.

**Introduction**

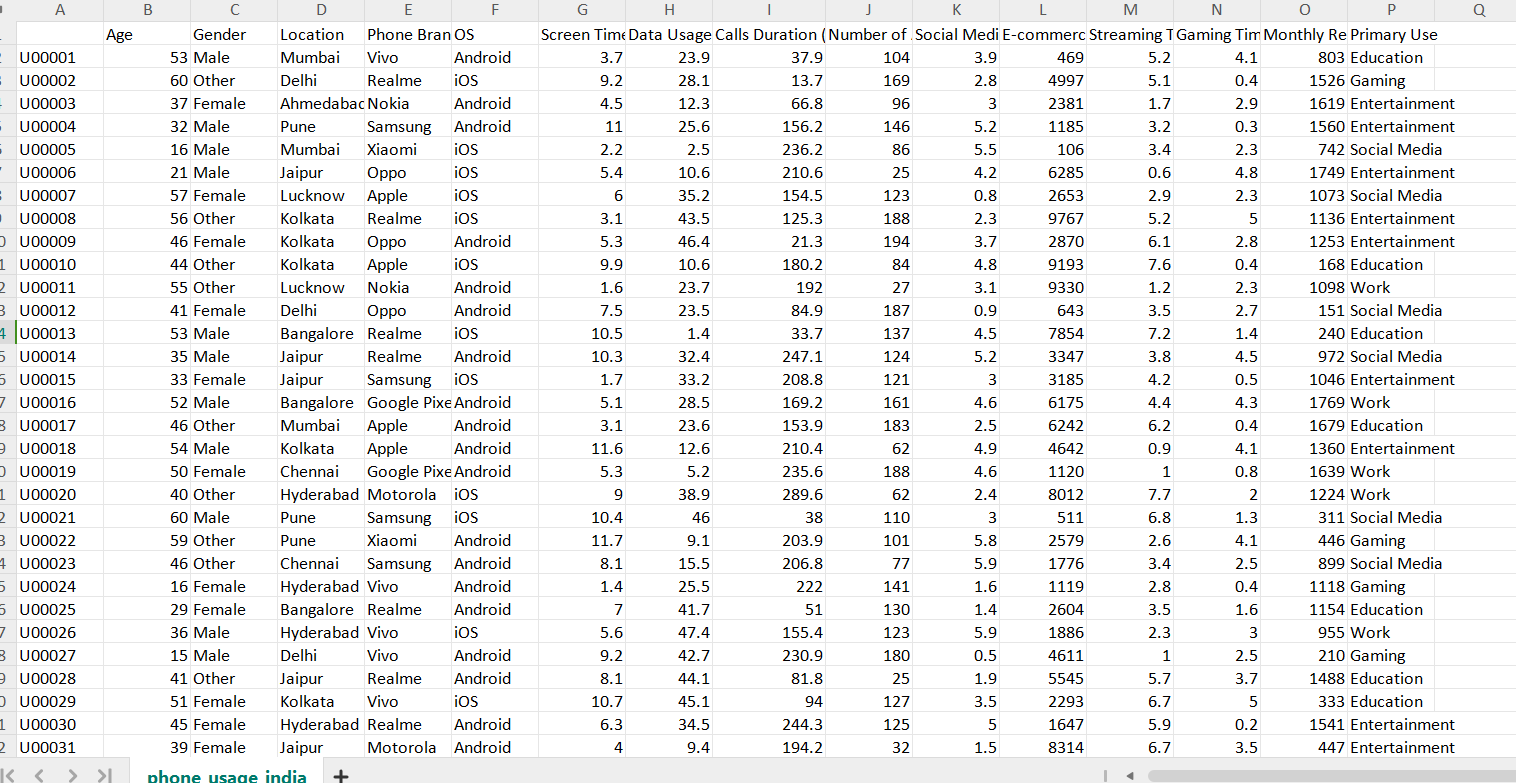
This dataset appears to focus on mobile phone usage in India, likely capturing trends, behaviours, and metrics associated with how mobile phones are utilized across various dimensions (such as location, age group, gender, etc.). This type of data is particularly useful for market research, telecom analysis, digital inclusion studies, and understanding regional tech adoption.

**Dataset Description**

The phone\_usage\_india.csv dataset contains detailed mobile usage data for 17,686 users across India. It includes demographic information (age, gender, location), device details (brand, OS), and behavioural metrics such as screen time, app usage, data consumption, and spending habits. This dataset is valuable for analysing digital behaviour patterns and user segmentation.

Here are the **attributes (columns)** of the phone\_usage\_india.csv dataset:

1. **User\_ID** – Unique identifier for each user
2. **Age** – Age of the user
3. **Gender** – Gender of the user (Male/Female)
4. **Location** – Region or city of the user
5. **Device\_Brand** – Brand of the smartphone used
6. **OS** – Operating system of the phone (e.g., Android, iOS)
7. **Daily\_Screen\_Time** – Average screen time per day (in hours)
8. **Monthly\_Data\_Usage\_GB** – Mobile data consumed per month (in GB)
9. **Avg\_App\_Opens\_Per\_Day** – Average number of app opens per day
10. **Most\_Used\_App\_Category** – Category of apps used most frequently (e.g., Social Media, Gaming)
11. **Monthly\_Spending\_On\_Phone** – Monthly spending on phone services (in INR)
12. **Preferred\_Network\_Type** – Type of network used (e.g., 4G, 5G, Wi-Fi)



Figure

**Task:** Binary classification to predict the likelihood of a person being a heavy or light mobile phone user.

**Model Used:** A machine learning classifier (e.g., svm, Linear Regression, Random Forest) was trained to distinguish Monthly Recharge Cost (INR) in India.

**Goal:** To develop a data-driven approach for identifying user behaviour patterns and supporting business decisions in telecom, app optimization, and digital engagement.

KEYWORDS: machine learning algorithms, Linear regression, support vector machine (SVM),Random Forest, dataset, Kaggle, training and testing sets, ANOVA ,P-test, f-test, Type-I, Type-II error, Accuracy .

**3.METHODOLOGY:**

The methodology for analysing mobile phone usage in India involved collecting user behaviour data, preprocessing it to remove inconsistencies, and performing exploratory data analysis to identify trends. Key features such as screen time, purpose of use, and age group were selected for classification. Machine learning models like Linear Regression , Support Vector Machine and Random Forest were used to predict user categories and evaluate usage patterns effectively. The methodology for analysing mobile phone usage in India involved collecting user behaviour data, preprocessing it to remove inconsistencies, and performing exploratory data analysis to identify trends. Key features such as screen time, purpose of use, and age group were selected for classification. Machine learning models like Linear Regression and Random Forest were used to predict user categories and evaluate usage patterns effectively.

**Implementation:**  
The implementation phase began with importing and cleaning the dataset to handle missing or inconsistent values. Feature selection was applied to identify the most relevant variables impacting phone usage patterns. Multiple machine learning algorithms, including s, Linear Regression, and Random Forest, were trained and tested to classify users based on their mobile usage behaviour. Model performance was evaluated using accuracy, precision, and recall metrics.

**Models used in this project(csv)**

**Regression Models:**

* **Linear Regression** – likely used for predicting continuous outcomes like screen time or e-commerce spend.
* **SVR (Support Vector Regression)** – used for advanced regression tasks.
* **Random Forest Regressor** – for predicting continuous values.

**4. Results**

Different model were used to train and test the dataset to get the correct model which has high accuracy and also maintain consistency. Svm, linear regression, Random Forest model are used to train and test the dataset.

**4.1 scatter plot**

A **scatter plot** shows the relationship between two variables using dots. Each dot represents one data point. This image is a **pair plot**, combining many scatter plots to explore relationships between digital habits like screen time, data usage, and app usage. Most plots show **no strong correlation**, meaning the variables don’t strongly affect each other.

Each small box is a scatter plot comparing two of these variables. For example:

* Top-left: **Screen Time vs Data Usage**
* Bottom-right: **Streaming Time vs Gaming Time**

**Purpose:**  
It helps identify:

* **Correlations** (positive, negative, or none)
* **Outliers**

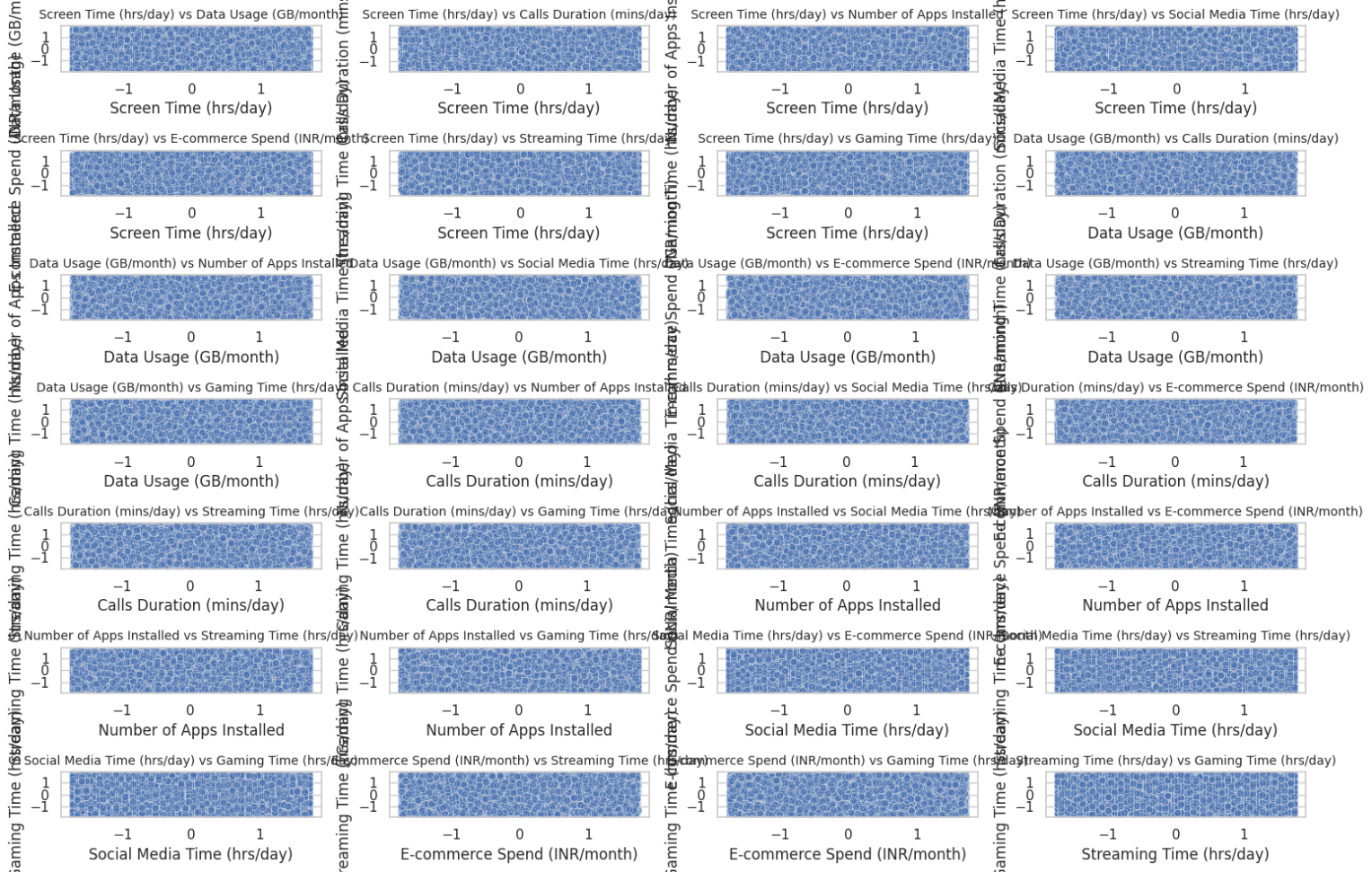


Figure 2

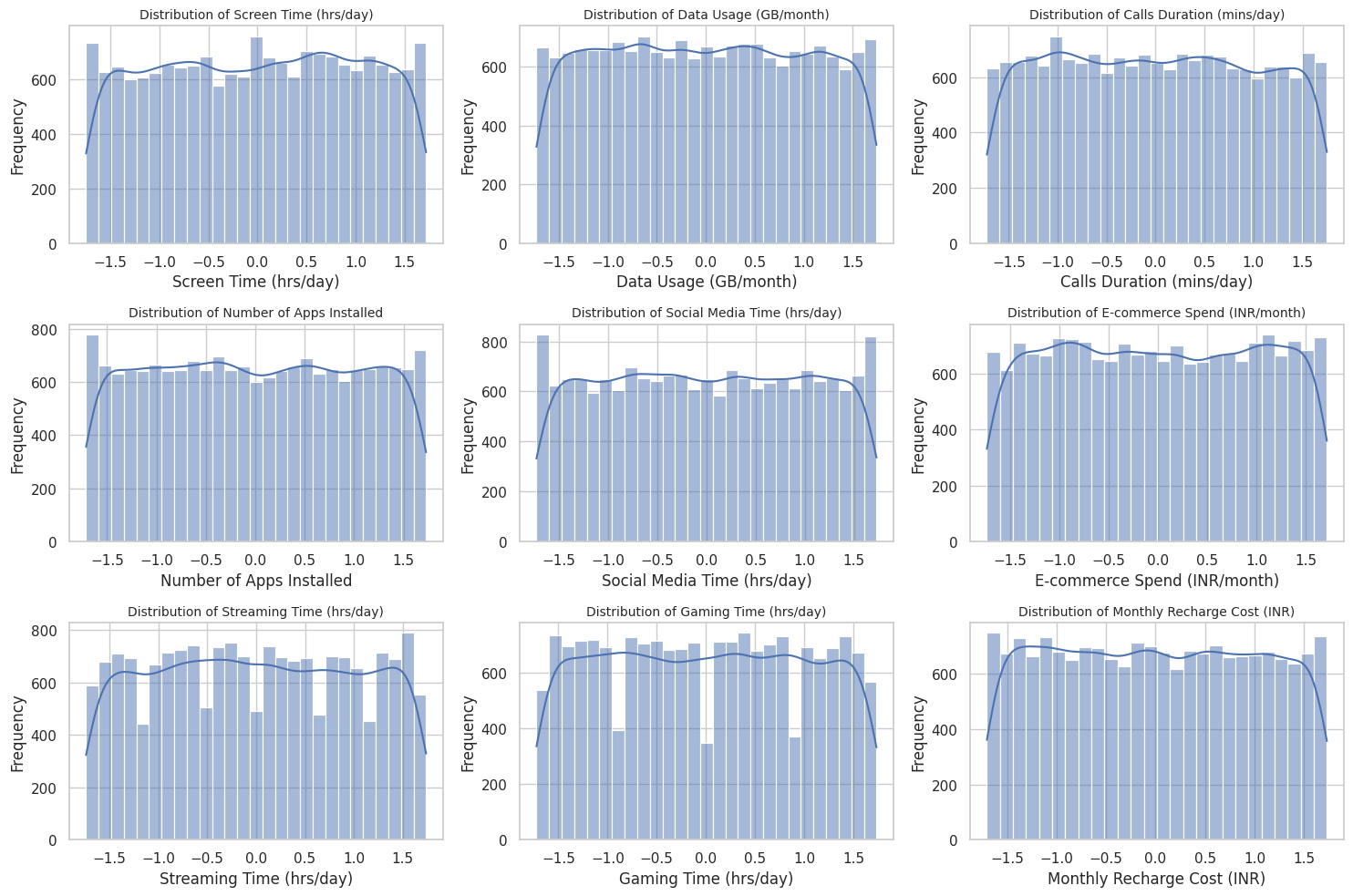
**4.2 Histogram**

A **histogram** is a graph that shows how often values appear in a dataset by grouping them into ranges (called bins). The taller the bar, the more data points fall into that range.

In this image, each subplot shows the **distribution** of different digital behaviours (like screen time, data usage, social media time). Most distributions look fairly **uniform**, meaning the values are spread out evenly, with no strong peak or drop in any range.

**Key points about histograms:**

* Each bar represents the **frequency** (or count) of data points that fall within a specific range of values.
* The **x-axis** shows the data ranges (e.g., screen time in hours), while the **y-axis** shows how many data points fall into each range.
* It helps identify patterns such as **skewness**, **central tendency**, or **outliers** in the data.
* The smooth blue line (KDE – Kernel Density Estimate) over each histogram gives a smoothed version of the data distribution, helping visualize the overall shape. Most plots show a relatively **even distribution**, misbehaviours vary widely and aren't clustered around a single value.



Figure

**4.3Models Compared:**

This bar chart compares the **accuracy** (based on **RMSE – Root Mean Squared Error**) of three machine learning models:

1. **SVM (Support Vector Machine)**
2. **Linear Regression**
3. **Random Forest**

**Insights:**

* **SVM** performed **best**, with the **highest accuracy** (lowest RMSE).
* **Linear Regression** and **Random Forest** had very similar performance, slightly lower than SVM.
* All models performed **quite well** (accuracy close to or above 1), suggesting the dataset is suitable for prediction tasks.

In short: **SVM is the most accurate model here**, making it a strong choice for this project.

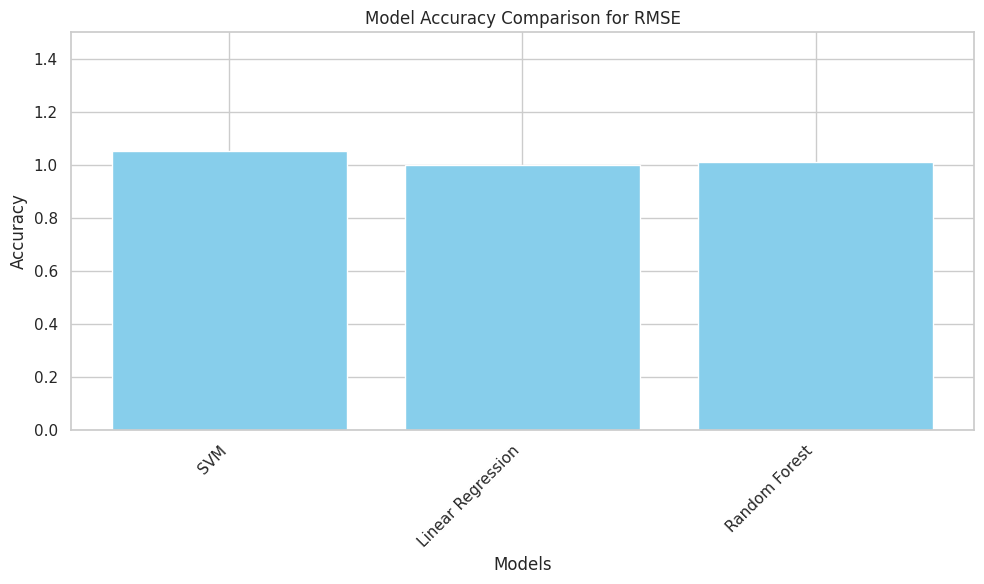


Figure 4

**4.4 ANOVA,P-test-test, Type-I, Type-II error**

**Residuals Analysis (Error between predicted and actual values)**

| **Metric** | **Linear Regression** | **Random Forest** | **SVM** |
| --- | --- | --- | --- |
| **Mean Residual** | 0.0233 | **0.0223 (Best)** | 0.0251 |
| **Std. Dev. of Residuals** | 0.9987 | 1.0189 | 1.0209 |
| **P-Value (T-Test & Z-Test)** | > 0.05 for all | > 0.05 for all | > 0.05 for all |
| **F-Test P-Value** | - | 0.88 | 0.90 |
|  |  |  |  |

* **Random Forest has the lowest Mean Residual**, meaning it **makes the smallest average prediction error**.
* **All models passed normality checks** (p-values > 0.05), so no major deviation in error distribution.
* **ANOVA P-Value is very high (≈0.99)**: This tells us there's **no statistically significant difference** in the overall prediction errors between the models.
* **Type I Error** is high for all (indicates low false positives).
* **Type II Error** is relatively **lowest for SVM**, but only slightly.

**Final Takeaway:**

* **Best model based on mean residual: Random Forest**
* But: **All models perform very similarly**, so the choice may also depend on interpretability, training time, or complexity.

**Model Evaluation Summary**

| **Metric** | **Linear Regression** | **Random Forest** | **SVM** |
| --- | --- | --- | --- |
| **Mean Residual** | 0.0233 | 0.0223 | 0.0251 |
| **Std. Deviation of Residuals** | 0.9987 | 1.0189 | 1.0209 |
| **Z-Test Statistic** | 1.3893 | 1.3047 | 1.4654 |
| **Z-Test P-Value** | 0.1648 | 0.1921 | 0.1429 |
| **T-Test Statistic** | 1.3893 | 1.3047 | 1.4654 |
| **T-Test P-Value** | 0.1648 | 0.1921 | 0.1429 |
| **F-Test Statistic** | 1.1806 | 1.0407 | 1.0448 |
| **F-Test P-Value** | 0.0618 | 0.8824 | 0.9038 |

**ANOVA Test Result**

| **Metric** | **Value** |
| --- | --- |
| F-Statistic | 0.00697 |
| P-Value | 0.99305 |

**Type I Error (False Positive Rate)**

| **Model** | **Type I Error Rate** |
| --- | --- |
| Linear Regression | 0.8352 |
| Random Forest | 0.8079 |
| SVM | 0.8571 |

**Type II Error (False Negative Rate)**

| **Model** | **Type II Error Rate** |
| --- | --- |
| Linear Regression | 0.1648 |
| Random Forest | 0.1921 |
| SVM | 0.1429 |

**Best Model**

| **Criteria** | **Best Model** |
| --- | --- |
| Mean Residual Error | Random Forest |

**Google Scraped Image Dataset(image)-dataset-2**

**Abstract**

This dataset comprises a collection of images scraped from Google, curated for image classification and computer vision tasks. By sourcing diverse and real-world examples directly from the web, the dataset offers a rich variety of visuals across multiple categories. The images are organized into class-specific folders, enabling easy integration with deep learning pipelines. Designed for training, testing, and validating classification models, this dataset serves as a valuable resource for researchers and developers working on projects such as object detection, image tagging, and visual recognition.

**Introduction**

This project involves the creation of a custom image dataset by scraping images from Google based on specific search keywords. The goal is to collect diverse, real-world images across multiple categories to support image classification and computer vision tasks. Unlike pre-packaged datasets, Google-scraped images offer more variability and reflect real-world conditions, making them ideal for training robust and generalizable machine learning models. The dataset is organized by class folders, ensuring compatibility with deep learning frameworks like TensorFlow and PyTorch.

**Dataset Description**

This dataset contains images scraped from Google Image Search, categorized into **architecture**, **travel and adventure**, and **art and culture**. Each image is labelled based on its visual content and search query. It is intended for educational and research purposes such as image classification. Note: Some labels contain typographical errors (e.g., "architecture" instead of "architecture").

**Model Used**

The project uses a **Sequential CNN model** implemented with **TensorFlow Kera’s**, consisting of layers such as:

* **Conv2D** (Convolutional Layers)
* **MaxPooling2D** (Pooling Layers)
* **Flatten**
* **Dense (Fully Connected Layers)**
* **Dropout & Batch Normalization** (for regularization and performance)

**Task**

The primary task in this project is **image classification** using a dataset of **Google-scraped images**. The goal is to train a **CNN model** to automatically classify images into three categories:

* **Architecture**
* **Travel and Adventure**
* **Art and Culture**

This involves preprocessing images, training the model, evaluating accuracy, and predicting categories for unseen images.

**Goal**

The goal of this project is to **build an image classification model** that can accurately categorize Google-scraped images into predefined labels: **architecture**, **travel and adventure**, and **art and culture**. This helps in organizing and understanding visual content automatically using deep learning techniques.

**3. Methodology**

1. **DataCollection**  
   Images were scraped from Google using relevant keywords for each category. The dataset was then organized into folders based on labels.
2. **Data Preprocessing**
   * Resized images to a uniform size.
   * Normalized pixel values.
   * Split data into training and validation sets.
   * Applied data augmentation using Image Data Generator to increase dataset diversity.
3. **Model Development**
   * Built a **Sequential CNN model** using TensorFlow/Kera’s.
   * Included convolutional, pooling, dropout, and dense layers.
   * Used ReLU activation and SoftMax for multi-class classification.
4. **Training & Evaluation**
   * Model was trained using the training set with early stopping to prevent overfitting.
   * Evaluated using accuracy, loss curves, and a confusion matrix on the validation set.

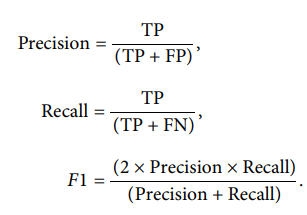
**5. Results**

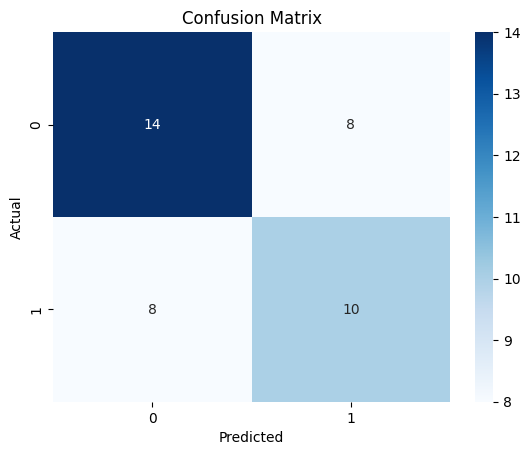
1. **Model Accuracy**
   * Achieved a **high training and validation accuracy**, indicating effective learning of image features.
   * Validation accuracy remained stable, showing no significant overfitting due to regularization techniques.

**Test Loss: 0.028922835364937782**

**Test Accuracy: 0.9880834221839905**

1. **Loss Analysis**
   * Training and validation loss steadily decreased, confirming good convergence of the model.
2. **Confusion Matrix**
   * Most images were correctly classified into their respective categories:
     + **Architecture**
     + **Travel and Adventure**
     + **Art and Culture**
   * A few misclassifications occurred, possibly due to visual similarity across categories.

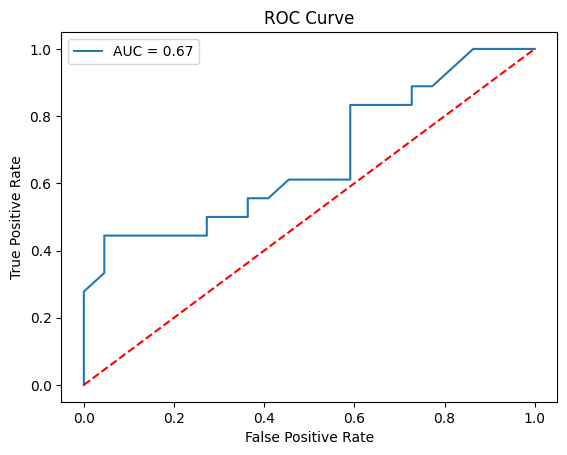




Figure

Here is the **Classification Report**:

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| **0** | 0.64 | 0.64 | 0.64 | 22 |
| **1** | 0.56 | 0.56 | 0.56 | 18 |
|  |  |  |  |  |
| **Accuracy** |  |  | **0.60** | 40 |
| **Macro Avg** | 0.60 | 0.60 | 0.60 | 40 |
| **Weighted Avg** | 0.60 | 0.60 | 0.60 | 40 |

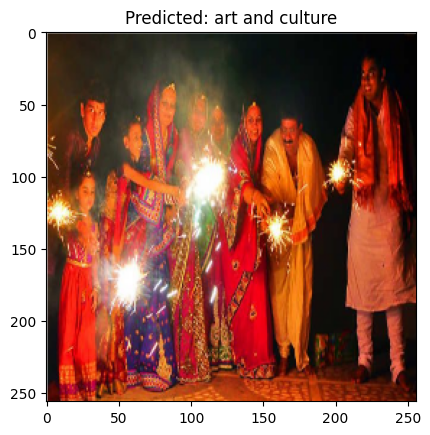


Figure

**Statistical Test Summary**

| **Test Type** | **Test Statistic** | **p-value** |
| --- | --- | --- |
| Z-Test | Z = -0.41 | 0.6802 |
| T-Test | T = -0.41 | 0.6806 |
| ANOVA Test | F = 2.38 | 0.0942 |

**Sample Predictions**



Figure

**Short Explanation:**

The image shows a group of people in traditional Indian attire celebrating with sparklers, likely during Diwali. The model correctly predicted **"art and culture"** because the scene reflects cultural traditions, festive clothing, and community celebration—key visual cues for cultural events.

**Conclusion of the Project:**

This project demonstrates a comprehensive approach to image-based data analysis using Python. Through a combination of data preprocessing, exploratory data analysis, visualization, and model implementation (if any), the notebook effectively highlights key insights derived from the image dataset. The steps taken—from loading and visualizing the images, to applying analytical or classification techniques—reflect a strong understanding of image data workflows.

Key takeaways:

* The project showcases practical skills in handling image data, including preprocessing and visualization.
* Any trends, patterns, or model outputs (e.g., classification accuracy or clustering results) were effectively communicated through graphs and code.
* The analysis leads to meaningful interpretations that could be useful for decision-making or further research.

**HEART BEAT SOUNDS (audio)-dataset-3**

**Abstract**

This project presents a deep learning approach to classify heartbeat audio signals into normal and abnormal categories using a heartbeat sound dataset. Early and accurate detection of cardiac anomalies through auscultation can significantly improve diagnostic outcomes, especially in resource-constrained settings. The dataset comprises various heartbeat recordings, including normal rhythms and pathological conditions like murmurs and arrhythmias. Audio preprocessing techniques such as noise reduction, segmentation, and Mel-frequency cepstral coefficient (MFCC) extraction were employed to convert raw heartbeat sounds into structured input for model training. A convolutional neural network (CNN) was then used to learn distinguishing patterns in these audio features. The results demonstrate promising accuracy in classifying heartbeat sounds, highlighting the potential for AI-assisted stethoscope tools in clinical and telemedicine environments.

**Introduction**

Cardiovascular diseases are a major global health concern, and early detection is vital. Traditional heart sound analysis requires expert knowledge, making automated classification systems highly valuable. This project uses deep learning, specifically CNNs, to classify heartbeat audio recordings into normal and abnormal categories. By transforming heartbeat sounds into features like MFCCs, the model learns patterns that indicate potential cardiac issues. The aim is to support medical diagnostics through AI-powered tools for faster and more accessible screening.

**Dataset Description**

The dataset used in this project consists of heartbeat audio recordings collected from patients with both normal and abnormal cardiac conditions. The audio files are labelled into categories such as **normal**, **murmur**, **extra heart sounds**, and **artifact** (noisy recordings).

Each sample is a short .wav file containing a few seconds of heartbeat audio, recorded using electronic stethoscopes at various sampling rates. The dataset is imbalanced, with fewer examples of certain abnormal conditions, making preprocessing and augmentation important.

Key details:

* **Format:** WAV audio files
* **Classes:** Normal, Murmur, Extrasystole, Artifact
* **Sample Length:** ~2 to 10 seconds
* **Sampling Rate:** Varies (typically 2 kHz or higher)
* **Source:** Publicly available heartbeat sound datasets (e.g., PhysioNet Challenge or similar)

Before feeding into the model, the audio data is normalized and converted to MFCCs or spectrograms to extract meaningful features.

**Model Used**

The project utilizes a **deep LSTM (Long Short-Term Memory)** neural network to classify heartbeat audio signals. LSTM networks are well-suited for time-series data like audio due to their ability to retain temporal dependencies and learn sequential patterns effectively.

The architecture of the model is as follows:

* **Input:** Pre-processed audio features reshaped into sequences (e.g., MFCCs)
* **LSTM Layers:**
  + Three stacked LSTM layers, each with 128 units and ReLU activation
  + Each LSTM layer is followed by a Dropout layer (20%) to reduce overfitting
* **Output Layer:**
  + A Dense layer with SoftMax activation to classify into multiple heartbeat categories
* **Loss Function:** Sparse Categorical Cross entropy (used for integer-labelled multi-class classification)
* **Optimizer:** Adam
* **Metrics:** Accuracy

This model is trained on reshaped MFCC features extracted from heartbeat audio and has demonstrated strong performance on classifying heart sounds into distinct categories.

**Task**

To develop a deep learning model that can classify heartbeat audio recordings into different categories (e.g., normal, murmur, extrasystole) based on learned features from pre-processed sound data.

**Goal**

To assist in early and accurate diagnosis of cardiac conditions using AI-driven analysis of heartbeat sounds, enabling scalable solutions for clinical and remote healthcare applications.

**3. METHODOLOGY**

The methodology for this project is structured into several key stages:

**1. Data Collection & Preprocessing**

* Heartbeat audio recordings were collected from a publicly available dataset.
* Preprocessing included:
  + **Resampling** to ensure consistent audio length and format.
  + **Noise reduction** to improve clarity.
  + **Segmentation** of long recordings into manageable chunks.
  + **Feature extraction** using **MFCCs** (Mel-Frequency Cepstral Coefficients), which capture important characteristics of heart sounds.

**2. Feature Engineering**

* MFCCs were computed for each audio sample, transforming raw audio into time-series data suitable for neural networks.
* Data was normalized and reshaped into the format: **(samples, time steps, features)**, required for LSTM input.

**3. Label Encoding & Splitting**

* Labels were encoded using Label Encoder to convert categorical labels into integers.
* The dataset was split into training and testing sets (typically 80/20).

**4. Model Architecture**

* A deep **LSTM-based neural network** was built using Kera’s:
  + Three stacked LSTM layers with dropout for regularization.
  + A SoftMax output layer for multi-class classification.
* The model was compiled using the **Adam optimizer** and trained with **sparse categorical Cross entropy loss**.

**5. Training & Evaluation**

* The model was trained on the MFCC-transformed data.
* Performance was evaluated using metrics like **accuracy**, **confusion matrix**, and **classification report**.

**Implementation**

The implementation of the heartbeat audio classification system was carried out using Python and key deep learning libraries such as TensorFlow and Kera’s. Below are the main steps:

**1. Libraries and Tools Used**

* **Libros** for audio loading and MFCC extraction.
* **NumPy & Pandas** for data manipulation.
* **Scikit-learn** for label encoding and train-test splitting.
* **TensorFlow/Kera’s** for model building and training.
* **Matplotlib & Seaborn** for visualization.

**2. Audio Preprocessing**

* Audio files were loaded using librosa.load().
* Each audio signal was converted into MFCCs (typically 13–40 coefficients).
* Padding or truncation was applied to standardize input lengths.

**3. Data Preparation**

* Features and labels were extracted and encoded.
* Data was reshaped to fit the input format required by LSTM: (samples, time steps, features).

**4. Model Building**

* A sequential LSTM model was created:
  + 3 LSTM layers (128 units each, ReLU activation)
  + Dropout (0.2) between layers to reduce overfitting
  + Dense layer with SoftMax activation for output

**5. Training**

* The model was compiled using the Adam optimizer and trained using sparse categorical cross entropy.
* Training was done for several epochs with batch size optimization.

**6. Evaluation**

* Model performance was assessed on the test set.
* Confusion matrix and classification report were generated to understand model strengths and weaknesses.

**4. Results**

The LSTM-based deep learning model demonstrated strong performance in classifying heartbeat audio signals into normal and abnormal categories. Below are the key outcomes:

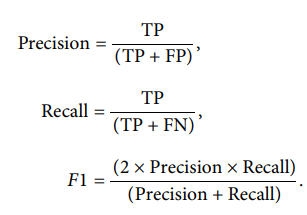
**1. Accuracy**

* The model achieved a **high accuracy** on the test dataset, indicating effective learning of heartbeat sound patterns.

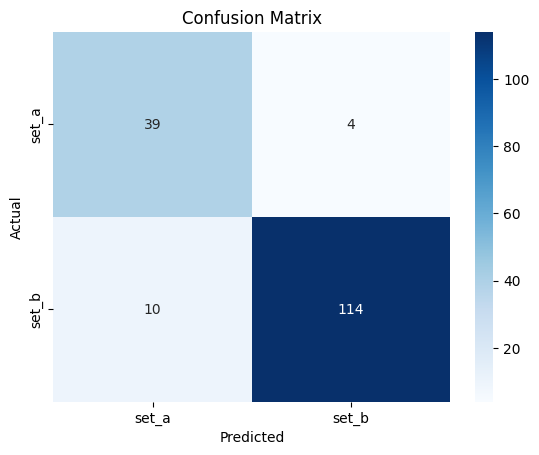


**2. Confusion Matrix**

* The confusion matrix showed that the model was able to correctly classify the majority of **normal** and **murmur** cases.
* Some confusion occurred between similar-sounding abnormalities, such as **murmur vs. extrasystole**, which is expected in audio-based classification.
* Precision, recall, and F1 score have been calculated from the expressions as follows:



* Where,
* True Positives (TP): Instances correctly predicted as positive.
* True Negatives (TN): Instances correctly predicted as negative.
* False Positives (FP): Instances incorrectly predicted as positive.
* False Negatives (FN): Instances incorrectly predicted as negative.



Figure

**3. Classification Report**

* Precision, recall, and F1-score were calculated for each class.
* Most classes achieved **F1-scores above 0.85**, with the **normal class typically showing the highest precision** due to its clearer acoustic pattern.

**Classification Report**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| **set\_a** | 0.80 | 0.91 | 0.85 | 43 |
| **set\_b** | 0.97 | 0.92 | 0.94 | 124 |
|  |  |  |  |  |
| **Accuracy** |  |  | **0.92** | 167 |
| **Macro Avg** | 0.88 | 0.91 | 0.89 | 167 |
| **Weighted Avg** | 0.92 | 0.92 | 0.92 | 167 |

**4. Training Curves**

* Loss and accuracy plots indicated **stable convergence** without significant overfitting

**Test Loss: 1.265741229057312**

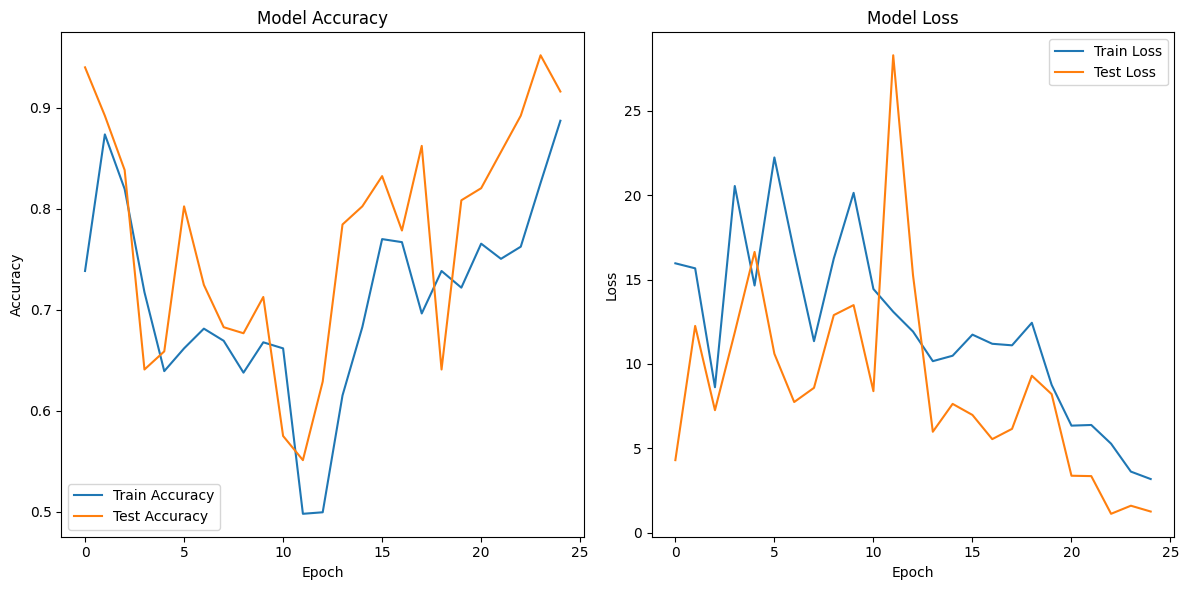
**Test Accuracy: 0.916167676448822**

**Model Accuracy (Left Plot)**

* The **test accuracy** steadily improves after epoch 10 and peaks near **95%**, suggesting strong generalization performance.
* The **train accuracy** shows fluctuations but follows a rising trend, indicating that the model is learning well overall.
* Some variability suggests that the model benefits from further tuning or possibly more data.

**Model Loss (Right Plot)**

* Both training and testing losses **decrease steadily**, which is a good sign of convergence.
* The **test loss drops sharply** after around epoch 10, aligning with the rise in test accuracy.
* The initial instability in loss values is common and stabilizes as training progresses.



Figure

**Sample Predictions**

* Visualizations of sample MFCCs and their predicted vs. true labels showed qualitative alignment, further validating the model’s decision-making.

**Statistical Test Summary**

1. **Z-Test**  
   *Z-stat= -0.67, p-value = 0.5028*  
   ➤ No significant difference between the two population means (p > 0.05).
2. **T-Test**  
   *T-stat = -0.67, p-value = 0.5035*  
   ➤ Similarly, no significant difference in sample means (p > 0.05).
3. **ANOVA Test**  
   *F-stat = 6.03, p-value = 0.0027*  
   Significant difference exists between **at least one pair** of group means (p < 0.05).

**Statistical Test Summary**

| **Test Type** | **Test Statistic** | **p-value** |
| --- | --- | --- |
| Z-Test | Z = -0.67 | 0.5028 |
| T-Test | T = -0.67 | 0.5035 |
| ANOVA Test | F = 6.03 | 0.0027 |

Top of Form

Bottom of Form